Improving IR Performance from OCRed Text using Cooccurrence

Kripabandhu Ghosh, Anirban Chakraborty and Swapan Kumar Parui
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute, Kolkata, West Bengal, India
{kripa.ghosh, chakraborty.abhi89, swapan.parui}@gmail.com

ABSTRACT

Information Retrieval performance is hurt to a great extent by OCR errors. Much research has been reported on modelling and correction of OCR errors. However, all the existing systems make use of language dependent resources or training texts to study the nature of errors. No research has been reported on improving retrieval performance from erroneous text when no training data is available. We propose a novel algorithm for automatic detection of OCR errors and improvement of retrieval performance from the erroneous corpus. Our algorithm does not use any training data or any language specific resources like thesaurus. It also does not use any knowledge about the language except that the word delimiter is blank space. We have tested our algorithm on erroneous OCRed Bangla FIRE collection offered in the RISOT 2012 track and obtained about 9% improvement over the OCRed baseline. However, the improvement is not statistically significant.

Categories and Subject Descriptors

H.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval

General Terms

Noisy corpus

Keywords

Cooccurrence, Query Expansion

1. INTRODUCTION

Erroneous text collections have presented research challenges to the scientists. Many such collections have been created. The researchers have tried error modelling and correcting techniques on them. Such techniques are on training models on sample pairs of correct and erroneous variants. But this is possible only if the training samples are available. There are several text collections which are created directly by scanning hard copies and then OCRing them. We are in an age of digitization. A large number of hard-copied documents are scanned and archived online. The Million Book Project[1] [2] was a book digitization project led by Carnegie Mellon University School of Computer Science and University Library. It was aimed at scanning and OCRing books of different languages. By December 2007, more than 1.5 million books have been scanned in 20 languages; mostly in Chinese, English, Telugu and Arabic. Indian Institute of Science, Bangalore collaborated with CMU in the Million Book Project[3]. More than 289,000 books have been scanned. Out of these nearly 170,000 are in Indian languages[1]. More than 84,000 books (25 million pages) are available on the DLI web site hosted by the Indian Institute of Science (http://www.new.dli.ernet.in). More than 149,000 books (43 million pages) are available on the DLI web site, which is hosted by the International Institute of Information Technology (http://dli.iiit.ac.in). The link to other partner sites are also provided through (http://www.new.dli.ernet.in). Contents between the two sites overlap in order to ensure fail safe availability. Another class of vital documents is comprized by the legal documents. Millions of court documents, defense documents, proprietary and legacy documents are in hard-copy format. The number of such documents is alarming in countries like India where they are in several Indian languages. Scanning, OCRing and archiving such volumes of documents are a great challenge itself. Information retrieval from such collections is no less a challenge since OCRs in Indian languages are not well-developed. Moreover, the print quality, font diversity and several other features of the hard-copies contribute heavily to the low quality of the scanned documents. Therefore, the clean, error-free version of such a collection is not available to be used for training purpose. Hence, error modelling on such datasets would require manual creation of the error-free version. ACM SIGIR Digital Museum[4] have archived lecture notes of IR stalwarts like Cyril W. Cleverdon, Gerard Salton, Joseph John Rocchio, K. Sparck Jones, etc as pdf versions created by scanning the original hard-copies of the same. This collection also lacks the original text version. OCRing of this collection is likely to generate erroneous texts which have to be corrected without the error-free version being available.

The unavailability of training data presents a different paradigm

FIRE '13, December 04 - 06 2013, New Delhi, India
Copyright 2013 ACM 978-1-4503-2830-2/13/12$15.00.
http://dx.doi.org/10.1145/2701336.2701648

[1]http://en.wikipedia.org/wiki/Million_Book_Project
[2]http://medlibrary.org/medwiki/Million_Book_Project
[3]http://deity.gov.in/content/national-digital-library
[4]www.sigir.org/museum/allcontents.html

altogether. In this paper we have made an endeavour of addressing this problem and proposing a possible solution to the problem. We have developed an algorithm based on word similarity and context information. A string matching technique (e.g., edit distance, n-gram overlaps, etc.) alone is not reliable in finding the erroneous variants of an error-free word due to homonymy. For example, word pairs like industrial and industrious, Kashmir (place) and Kashmira (name), etc. have very high string similarity and yet they are unrelated. Such mistakes are even so likely when we do not have a parallel error-free text collection to match the erroneous variants with the correct ones using the common context. However, context information can be used to get more reliable groups of erroneous variants. Context information can be harnessed effectively by word cooccurrence. We say that two words cooccur if they occur in a window of certain words between each other. Cancho et al. [8] focussed on the importance of word co-occurrences in capturing relationships among words in a human language network. Word cooccurrence have been used successfully in identifying better stems [14], [15] than methods that use string similarity only [12].

The rest of the paper is organised as follows:

In section 2, we discuss the related works. In section 3, we describe our method. We present the results in section 4. We have a discussion in section 5. We conclude in section 6.

2. RELATED WORK

Works have been reported on retrieval from OCRed text. Taghva et al. [20] applied probabilistic IR on OCRed text. Error correction was done using a domain-specific dictionary. The misspelt words were clustered around correctly spelt words, which were identified using the dictionary. If a cluster of misspelt words was close to more than one correctly-spelt word, the error patterns of the OCR used were analysed. Singhal et al. [18] reported that the linear document normalization models were better suited to collections containing OCR errors than the quadratic (cosine normalization) models. TREC took an important initiative on the study and effect of OCR errors in retrieval in their two tasks : the Confusion Track and the Legal Track. The TREC Confusion track was a part of TREC 4 (1995) [7] and TREC 5 (1996) [9]. In TREC 4 Confusion Track, random character insertions, deletions and substitutions were used to model degradations. Degradations were incorporated on 260,000 English electronic text documents from multiple sources. For the TREC 5 Confusion Track, 55,000 government announcement documents were printed, scanned, OCRed and then were used. Electronic text for the same documents was available for comparison. Participants experimented with techniques that used error modelling to handle OCR errors using character n-gram matches.

A similar track, RISOT [5], was offered in Forum for Information Retrieval Evaluation[5] (FIRE) 2011. The purpose of this task was to improve retrieval performance from OCRed text in Indic script. In 2011, a FIRE Bangla collection of 62,825 documents was available as the “TEXT” or “clean” collection from the Bangla newspaper, Anandabazar Patrika. Each document of the collection was scanned at a resolution of 300 dots per inch; each scanned document was converted to electronic text using a Bangla OCR system that had about 92.5% accuracy. Ghosh et al. [6] employed a two-fold error modelling technique for OCR errors in Bangla script. In 2012 RISOT, a Hindi collection pair was also offered. The error-free Hindi document collection is created from the Hindi newspapers Dainik Jagaran and Amar Ujala. The OCRed Hindi collection was prepared using a Hindi OCR system which also had 92.5% accuracy.

Substantial work has been reported in the literature on OCR error modelling and correction. Kolak and Resnik [10] employed a pattern recognition approach in detecting OCR errors. Walid and Kareem [11] used Character Segment Correction, Language modelling, and Shallow Morphology techniques in error correction on OCRed Arabic texts. B.B. Chaudhuri and U. Pal produced the very first report on error detection and correction of Indic scripts in 1996 [2]. This paper used morphological parsing to detect and correct OCR errors. Separate lexicons of root-words and suffixes were used. Reynaert [16] has developed an online processing system for post-processing of OCR errors. It first derives the alphabet for the language from an appropriate source. Then, the valid characters for a given language are retained. Then, the list of all possible character confusions are produced according to a given thereshold for Levenshtein distance. Choudhury et al. [3] explored the challenges in developing a spell-checker orthographic proximity between two words for Bengali, English and Hindi.

3. OUR APPROACH

In this section we describe our approach. Most of the prior works on the identification of erroneous variants of a word have relied on string similarity alone. In this work, we have combined context information along with a string similarity measure to get more reliable erroneous variants. Before going into the algorithm, we describe the key concepts used in the algorithm in the following subsection. Then, we describe the algorithm in details.

3.1 Key Terms

3.1.1 Word cooccurrence

We say that two words, say, w_1 and w_2, cooccur if they appear in a window of size s ($s > 0$) words in the same document d. Suppose, the words w_1 and w_2 cooccur in a window of size 5 in a document d. This means that there is at least one instance in the document where at most 4 words (distinct from w_1 and w_2) occur between w_1 and w_2 or between w_2 and w_1. Let $cooccurFreq_{(d,s)}(w_1, w_2)$ denote the number of times w_1 and w_2 cooccur in d in a window of size s. Then, we call $cooccurFreq_{(d,s)}(w_1, w_2)$ the cooccurrence frequency of w_1 and w_2 in document d for a window of size s. However, it is a common practice to calculate $cooccurFreq_{(d,s)}(w_1, w_2)$ over all the documents in a collection. This is likely to give a more robust measure of co-location of the words w_1 and w_2.

Word cooccurrence gives a reliable measure of association between two words as it reflects the degree of context match between the two words. Usually, the total cooccurrence between word pairs is calculated over a collection of documents

[5]www.isical.ac.in/∼fire

by summing up the document-wise cooccurrence frequencies. High cooccurrence between a pair of words is an indicator of high degree of relatedness of the words. This association measure gets more strength when it is used in conjunction with a string matching measure. For example, two words sharing a long stem (prefix) is likely to be variants of each other if they share the same context as indicated by a high cooccurrence value between them. The word *industrious* shares a stem "industri" with the word *industrial*. But, they are not variants of each other. They can be easily segregated by examining their context match as they are unlikely to have a high cooccurrence frequency. In this paper, we have used cooccurrence information with a string similarity measure (LCS, discussed in the following subsection) to identify erroneous variants of query words.

3.1.2 Longest Common Subsequence (LCS) similarity

We choose Longest Common Subsequence (LCS) similarity as the string similarity measure in our algorithm. We could have also used the commonly used similarity measures like edit distance for the same purpose. But, we choose LCS because it has not been used before and we wanted to explore the utility of LCS similarity in our problem. But, prefix matching is not suitable in this case because an error can occur at any position of a word. For example, suppose a word *tobacco* has been misrecognised as *1obacco*. That is, the first 't' has been misrecognised as '0'. However, in this situation the prefix match is null although *1obacco* is a variant of *tobacco*.

Longest Common Subsequence [4] is defined as follows:

Given a sequence $X = \langle x_1, x_2, ..., x_m \rangle$, another sequence $Z = \langle z_1, z_2, ..., z_k \rangle$ is a *subsequence* of X if there exists a strictly increasing sequence $\langle i_1, i_2, ..., i_k \rangle$ of indices of X such that for all $j = 1,2,...,k$, we have $x_{i_j} = z_j$. Now, given two sequences X and Y, we say that Z is a *common subsequence* of X and Y if Z is a subsequence of both X and Y. A *common subsequence* of X and Y that has the longest possible length is called a *longest common subsequence* or LCS of X and Y. For example, let $X = \langle A, B, C, B, D, A, B \rangle$ and $Y = \langle B, D, C, A, B, A \rangle$. Then, the sequence $\langle B, D, A, B \rangle$ is the LCS of X and Y. In general, LCS of two sequences is not unique.

In our problem, we consider sequences of characters or strings. For strings *industry* and *industrial*, the LCS is *industr*. Now, we define a similarity measure as follows :

$$LCS\_similarity(w_1, w_2) = \frac{StringLength(LCS(w_1,w_2))}{Maximum(StringLength(w_1),StringLength(w_2))}$$

So, $LCS\_similarity(industry, industrial)$

$$= \frac{StringLength(LCS(industry,industrial))}{Maximum(StringLength(industry),StringLength(industrial))}$$
$$= \frac{StringLength(industr)}{Maximum(8,10)}$$
$$= \frac{7}{10}$$
$$= 0.7$$

Note that the value of LCS_similarity lies in the interval [0, 1].

3.1.3 Similar neighbours and dissimilar neighbours

We have used the notion of social relatedness as the pivotal point of our clustering algorithm. In this problem, we say that two words share the same neighbourhood of each other if they appear in the same document and have some degree of string similarity. Note that, we say that a string pair have "high similarity" if the similarity value is larger than some chosen threshold. However, there can be words which have high string similarity and are variants of each other. But, they do not cooccur in the same document. We attempt to bridge the relationship between these words using the words which are common neighbours. In other words, a word w is a *common neighbour* of words w_1 and w_2 if w cooccurs with w_1 and w_2 in different documents but w_1 and w_2 do not cooccur with each other. For this purpose, we have considered two degrees of relatedness between two words. First we consider that two words have a common neighbour. These two words are *similar neighbours* of each other if they also have high string similarity. *Similar neighbours* can be formally defined as:

Words w_1 and w_2 are *similar neighbours* if they are connected by a *common neighbour* and they have high string similarity with each other as well as with the *common neighbour*.

This situation is shown in Figure 1. Here *Tobacco* and *Tobacc0* share a common neighbour *Tobacc1*. Hence, *Tobacco* and *Tobacc0* become *similar neighbours* as they have high string similarity. Secondly, we say that two words are *dissimilar neighbours* if they have a common neighbour which has low string similarity with both. However, these two words are highly similar and are connected by the dissimilar word. We give a more formal definition below :

Words w_1 and w_2 are *dissimilar neighbours* if they are connected by a *common neighbour* and they have high string similarity with each other but not with the *common neighbour*.

This situation is shown in Figure 2. Here, *Tobacco* and *Tobacc* are connected by *Cigarette*. Also, *Tobacco* and *Tobacc* have high string similarity. So, here *Tobacco* and *Tobacc* are *dissimilar neighbours*. In Figure 1, we say that *Tobacc1*, *Tobac* and *obacc* are *close neighbours* of *Tobacco* since they appear in the same document with *Tobacco* and have high string similarity with *Tobacco*. However, *Cigarette*, *Smoking* and *Cancer* are not close neighbours of *Tobacco* because they only cooccur with *Tobacco* but are not erroneous variants of *Tobacco*.

3.2 The Proposed Approach

Our approach has two basic steps :

1. Clustering
2. Cluster selection

These steps are discussed in detail as follows :

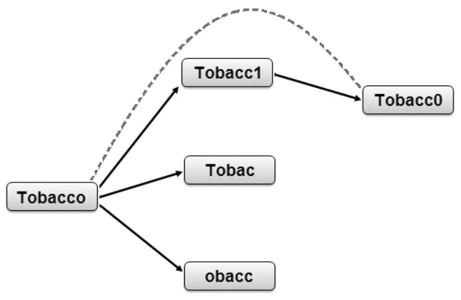


Figure 1: Similar neighbours

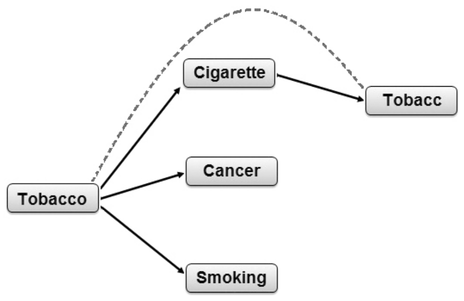


Figure 2: Dissimilar neighbours

1. **Clustering Algorithm**

   In Algorithm 1, we start with the set L of all unique words in the OCRed corpus. Here step 5 gets the *close neighbours* of w. We refer Figure 1. For the word *Tobacco*, {*Tobacc1*, *Tobac* and *obacc*} are the *close neighbours*. Step 6 gets the *similar neighbours* of the *close neighbours* obtained in the previous step. *Tobacc0* is one *similar neighbour* of *Tobacco* obtained through the *common neighbour Tobacc1*. So, at the end of step 7, $S_{closesimilar}$ contains all the *close neighbours* and *similar neighbours* of w. At step 9, we consider the neighbours (not necessarily the *close neighbours*) of w which have high cooccurrence with w. These neighbours do not necessarily have high similarity with w. In Figure 2, *Cigarette*, *Smoking* and *Cancer* are such neighbours of *Tobacco*. At step 11, we get the *dissimilar neighbours* of w. *Tobacc* is a *dissimilar neighbour* of *Tobacco*.

2. **Cluster Selection Algorithm**

   Next, we map the query words with the appropriate clusters. By the appropriate cluster of a query word w_q, we mean the cluster containing words of the OCRed corpus that contain the *variants* of w_q. Algorithm 2 shows our Cluster Selection algorithm.

   Note that Algorithm 1 clusters the whole OCRed corpus. We need to select appropriate clusters for each query word for expansion. In Algorithm 2, we do this selection process. At steps 2 and 3, for each query word, we identify the clusters which has at least one word that has high LCS similarity with the query word w_q. For example, let *Tobacco* be a query word. Let C = {*Tobacc*, *Tobacc1*, *Tobac*, *obac*} be a selected cluster. Then, at step 4, we construct $C' = C \cup \{w_q\}$ = {*Tobacco*, *Tobacc*, *Tobacc1*, *Tobac*, *obac*}. Note that C' contains the query word *Tobacco*. At step 5, we cluster C' using complete-linkage algorithm. From the resulting clusters, we keep only the one containing the query word and ignore the rest. Let C'' = {*Tobacco*, *Tobacc*, *Tobacc1*} be the cluster thus chosen. Now, for the query word, several clusters can be chosen by steps 2 and 3. Let these clusters be $C_1, C_2, \cdots, C_n$. After augmenting the query word, we get $C'_1, C'_2, \cdots, C'_n$. After clustering each C'_i using complete-linkage algorithm and eliminating the clusters not containing the query word, we get $C''_1, C''_2, \cdots, C''_n$. Next, according to step 6, we calculate pairwise LCS similarity of the words in each cluster C''_i and calculate the Geometric Mean of the pairwise similarities. For the cluster {*Tobacco*, *Tobacc*, *Tobacc1*}, we calculate LCS similarity of 3 pairs and calculate the Geometric Mean[6] of these 3 values. Let the Geometric Means of pairwise LCS similarities of $C''_1, C''_2, \cdots, C''_n$ be $GM_1, GM_2, \cdots GM_n$ respectively. Let GM_{max} be the maximum of the values $GM_1, GM_2, \cdots GM_n$. Then, we choose the cluster, C_{final} with GM value GM_{max} as the expansion of w_q. If the number of such clusters is more than one, we take the union of all the clusters to get a single cluster C^+_{final}. So, we choose C^+_{final} as the expansion of w_q.

Algorithm 1 : Clustering algorithm

```
1: Let $L$ be the set of all unique words in the OCRed corpus.
2: for each word $w$ in $L$ do
3:     Let $S_{closesimilar}$ and $S_{dissimilar}$ be empty sets. Let $S_{all} = S_{closesimilar} \cup S_{dissimilar}$.
4:     /* Close and Similar neighbours */
5:     For word $w_1$ cooccurring with $w$, calculate LCS_similarity between $w$ and $w_1$. Store $w_1$ in $S_{closesimilar}$ if LCS_similarity($w$, $w_1$) $>$ some threshold $T$.
6:     For each $w'$ in $S_{closesimilar}$, find the words $w_2$ cooccurring with $w'$ such that LCS_similarity($w$, $w_2$) $> T$. Include all these words in $S_{closesimilar}$.
7:     Repeat step (6) until no new word is added to $S_{closesimilar}$.
8:     /* Dissimilar neighbours */
9:     Consider top m (in terms of frequency in corpus) words cooccurring with $w$.
10:    For each such word $w_3$, find the words $w_4$ cooccurring with $w_3$ such that LCS_similarity($w$, $w_4$) $> T$. Include all these words in $S_{dissimilar}$.
11:    Repeat step (10) until no new word is added to $S_{dissimilar}$.
12: end for
```

4. RESULTS

We tested our algorithm on FIRE RISOT[7] Bangla collection. The collection statistics can be seen in Table 1. *Bangla original* is the "clean" or error-free version created from Anandabazar Patrika. *Bangla OCRed* is the scanned-and-OCRed

[6]http://en.wikipedia.org/wiki/Geometric_mean
[7]http://www.isical.ac.in/∼ fire/data.html

Algorithm 2 : Cluster Selection algorithm

```
1: for each query word $w_q$ do
2:     calculate LCS_similarity($w_q$, $w$), where $w$ is a word in each cluster $S$ formed by Algorithm 1.
3:     Choose all the clusters for which LCS_similarity($w_q$, $w$) for at least one word $w$ in the cluster is greater than threshold $T$.
4:     For each cluster $C$ obtained from step (3) define $C' = C \cup \{w_q\}$
5:     Create complete-linkage clusters from each $C'$ of step (4) and keep the cluster containing $w_q$.
6:     For a cluster $C$, let us consider LCS_similarity between each pair of words in it. Let $GM_C$ denote the Geometric Mean of LCS_similarity of all the pairs. Then, compute $GM_C$ for each cluster given by step (5).
7:     Select the cluster $C$ with maximum $GM_C$ as the appropriate cluster for $w_q$. If the number of such clusters is more than one, we take the union of all the clusters to form a single cluster $C^+$.
8: end for
```

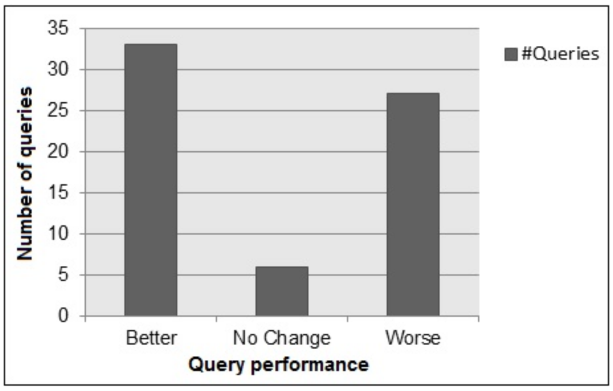
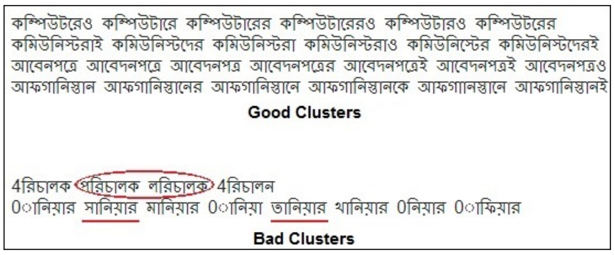


Figure 3: Query-wise performance

version of the same. A document in the original version and its OCRed version had the same unique document identification string so that the original-OCRed pairs can be easily identified. We can see that original version contains more documents than its OCRed version. So, naturally, the extra documents in the original could not be used for comparison. But, despite having fewer documents, we can see that the OCRed collection contains more unique terms than its error-free counterpart. The number of unique terms for the Bangla original corpus is 396968 while the same number for its OCRed version is 466867. This discrepancy is caused by OCR errors. Most of the inflations are caused by misrecognitions (as multiple candidates) and several other distortions. The Bangla collection has 66 topics. These topics were created for previous FIRE Ad Hoc tasks. A subset of the Ad Hoc topics were selected for the RISOT task.

Table 2 shows the results. The evaluation measures are Mean Average Precision (MAP) and Precision at 5 (P5) [13]. The experiments are done in Indri[8] toolkit. The Indri toolkit

[8]http://sourceforge.net/projects/lemur/



কম্পিউটরেও কম্পিউটারে কম্পিউটারের কম্পিউটারেরও কম্পিউটারও কম্পিউটরের
কমিউনিস্টরাই কমিউনিস্টদের কমিউনিস্টরা কমিউনিস্টরাও কমিউনিস্টের কমিউনিস্টদেরই
আবেনপত্রে আবেদনপত্রে আবেদনপত্র আবেদনপত্রের আবেদনপত্রেই আবেদনপত্রই আবেদনপত্রও
আফগানিস্তান আফগানিস্তানের আফগানিস্তানে আফগানিস্তানকে আফগানস্তানে আফগানিস্তানই

Good Clusters

4রিচালক পরিচালক লরিচালক 4রিচালন
0◌ানিয়ার সানিয়ার মানিয়ার 0◌ানিয়া তানিয়ার থানিয়ার 0নিয়ার 0◌াফিয়ার

Bad Clusters

Figure 4: Clusters

Dataset	No. of documents	No. of topics	No. of unique terms
Bangla original	62838	66	396968
Bangla OCRed	62825	66	466867

Table 1: Collection statistics

uses language model [19]. *Original text* is the result when all the queries are run on the "clean" or error-free version of the corpus. This value can be considered as an upper bound for performance. *OCRed text* is the result when the same set of queries are run on the OCRed version of the corpus. *RISOT2011* is the result produced by Ghosh et al. [6]. Note that *RISOT2011* uses the entire error-free version of the corpus for error modelling. *Proposed method* is the result produced by our method on the OCRed corpus. In the proposed method, we have three parameters, viz., T, m and the threshold for the complete linkage algorithm. In our experiments, we chose their values as 0.8, 10 and 0.6 respectively. We see that our method yields numerical improvement over the run *OCRed text* but this difference was found to be *not statistically significant* at 95% confidence level (p-value > 0.05) by Wilcoxon signed-rank test ([17]). The proposed approach is numerically inferior to *Original text* and *RISOT2011.*

5. ERROR ANALYSIS

We see that our method does not outperform the OCRed text baseline significantly. So, we made a query-wise analysis of our performance. We determined the number of queries for which the proposed method outperforms the OCRed baseline. There were 66 queries in total. Figure 3 shows that for for 33 queries only the proposed method is better than the OCRed baseline. For 27 queries, our method is worse than the baseline. There is no difference in performance for 6 queries. This indicates that our method leads to performance drop in about 41% of the cases. We then tried to inspect the reasons behind the dip. We looked at the clusters generated by our method. Figure 4 shows a se-

Run	MAP	P5
Original text	0.2567	0.3485
OCRed text	0.1791	0.2738
RISOT2011	0.2245	0.3098
Proposed method	0.1974	0.2831

Table 2: Results

lection of clusters. We show one cluster per line. The "Good Clusters" contain mostly correct inflectional variants of the query word. The four "Good Clusters" shown in the figure are the expanded versions of the words *computer*, *communist*, *abedanpatro* (Bangla of application) and *Afghanistan*. However, the "Bad clusters" reveal the possible reasons of performance fall. The figure shows two "Bad clusters". The first one is supposed to contain the variants of the word *porichalok* (Bangla of director). However, it also contains the word *lorichalok* which is the Bangla of lorry driver. So this cluster contains two different words which have high string similarity but are otherwise unrelated. The second cluster contains *sania* and *tania*, i.e., two different names. It also contains *maania*, Bangla of compliance. So, this is a highly heterogeneous cluster of unrelated terms. We have done a random sampling of the clusters and manually evaluated them. We have seen that about 10% of the clusters are "bad". This indicates we need to refine our algorithm to produce consistently uniform clusters that would contain words which are actual erroneous variants of each other rather than words which have high string similarity only and are weakly associated.

We note that one major drawback of our algorithm is the inability to eliminate the chance cooccurrences. It makes no effort to give more importance to the words which cooccur with each other in a large number of documents than the words that cooccur by chance in a few documents. The appearance of *porichalok* (Bangla of director) and *lorichalok* (Bangla of lorry driver) in the same cluster can be ascribed to chance cooccurrences. So, it is crucial that we attach importance to frequencies of cooccurrence. Moreover, we have to use a more restrictive clustering algorithm which would prevent loosely connected words from occurring in the same cluster.

6. CONCLUSION

In this paper we have proposed a new problem premise and made an endeavour to come up with a possible solution. We have shown that it is possible, to an extent, to improve retrieval performance from erroneous text even when the clean version is not available for error modelling. We see that harnessing the context information through word cooccurrence gives a reliable measure of grouping inflectional error variants. However, chance cooccurrences can be harmful as unrelated words can occur together in a small number of documents. So, we would aim at using a better clustering algorithm that would generate more compact and meaningful clusters. The proposed method can be of practical use as it can be used effectively to retrieve important information from collections which do not have error-free text version. The proposed approach is language-independent and so can be used across different text collections without language specific resources. So, in a nut shell, we look forward to alleviating the drawbacks of our method and develop a more effective version in future.

7. REFERENCES

[1] N. Balakrishnan. Digital library of india : A testbed for indian language research `http://www.ieee-tcdl.org/Bulletin/v3n1/balakrishnan/balakrishnan.html`. *TCDL Bulletin*, 3.

[2] B. Chaudhuri and U. Pal. Ocr error detection and correction of an inflectional indian language script. In *Proceedings of the 13th Int. Conf. on Pattern Recognition*, volume 3, pages 245–249, Vienna, 1996.

[3] M. Choudhury, M. Thomas, A. Mukherjee, A. Basu, and N. Ganguly. How difficult is it to develop a perfect spell-checker? a cross-linguistic analysis through complex network approach. *Proceedings of the second workshop on TextGraphs: Graph-based algorithms for natural language processing*, pages 81–88, 2007.

[4] T. Cormen, C. Leiserson, R. Rivest, and C.Stein. *Introduction to Algorithms.* The MIT Press, 3rd edition, 2009.

[5] U. Garain, J. Paik, T. Pal, P. Majumder, D. Doermann, and D. Oard. Overview of the fire 2011 risot task. volume 7536, pages 197–204. Springer, Oct. 2013.

[6] K. Ghosh and S. K. Parui. Retrieval from ocr text : Risot track. volume 7536, pages 214–226, Lecture Notes in Computer Science, Oct. 2013. Springer.

[7] D. Harman. Overview of the fourth text retrieval conference. pages 1–24. The Fourth Text Retrieval Conference, 1995.

[8] R. F. i Cancho and R. V. Solé. The small world of human language. *Proceedings of The Royal Society of London. Series B, Biological Sciences*, 268:2261–2265, 2001.

[9] P. Kantor and E. Voorhees. Report on the trec-5 confusion track. pages 65–74. The Fifth Text Retrieval Conference, 1996.

[10] O. Kolak and P. Resnik. Ocr error correction using a noisy channel model. pages 257–262. Proceedings of the Second International Conference on Human Language Technology Research, 2002.

[11] W. Magdy and K. Darwish. Arabic ocr error correction using character segment correction, language modeling, and shallow morphology. pages 408–414. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), 2006.

[12] P. Majumder, M. Mitra, S. K. Parui, G. Kole, P. Mitra, and K. Datta. Yass: Yet another suffix stripper. *ACM Trans. Inf. Syst.*, 25(4):18:1–18:20, Oct. 2007.

[13] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[14] J. H. Paik, M. Mitra, S. K. Parui, and K. Järvelin. Gras: An effective and efficient stemming algorithm for information retrieval. *ACM Trans. Inf. Syst.*, 29(4):19:1–19:24, Dec. 2011.

[15] J. H. Paik, D. Pal, and S. K. Parui. A novel corpus-based stemming algorithm using co-occurrence statistics. In *ACM SIGIR*, SIGIR '11, pages 863–872, 2011.

[16] M. Reynaert. Ticclops: Text-induced corpus clean-up as online processing system. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 52–56. Dublin City University and Association for Computational Linguistics, 2014.

[17] S. Siegel. *Nonparametric statistics for the behavioral*

sciences. McGraw-Hill series in psychology. McGraw-Hill, 1956.

[18] A. Singhal, G. Salton, and C. Buckley. Length normalization in degraded text collections. pages 149–162, Las Vegas, NV, USA, 1996. In Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval.

[19] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: a language-model based search engine for complex queries. Technical report, link : http://ciir.cs.umass.edu/pubfiles/ir-407.pdf, 2005.

[20] K. Taghva, J. Borsack, and A. Condit. Results of applying probabilistic ir to ocr text. pages 202–211, Dublin, Ireland, 1994. In The Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.